Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2004)
R. Boulic, D. K. Pai (Editors)

Point Based Animation of Elastic, Plastic and Melting Objects

M. Müller[1], R. Keiser[1], A. Nealen[2], M. Pauly[3], M. Gross[1] and M. Alexa[2]

[1] Computer Graphics Lab, ETH Zürich
[2] Discrete Geometric Modeling Group, TU Darmstadt
[3] Stanford University

Abstract

We present a method for modeling and animating a wide spectrum of volumetric objects, with material properties anywhere in the range from stiff elastic to highly plastic. Both the volume and the surface representation are point based, which allows arbitrarily large deviations form the original shape. In contrast to previous point based elasticity in computer graphics, our physical model is derived from continuum mechanics, which allows the specification of common material properties such as Young's Modulus and Poisson's Ratio.
In each step, we compute the spatial derivatives of the discrete displacement field using a Moving Least Squares (MLS) procedure. From these derivatives we obtain strains, stresses and elastic forces at each simulated point. We demonstrate how to solve the equations of motion based on these forces, with both explicit and implicit integration schemes. In addition, we propose techniques for modeling and animating a point-sampled surface that dynamically adapts to deformations of the underlying volumetric model.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Physically Based Modeling I.3.7 [Computer Graphics]: Animation and Virtual Reality

1. Introduction

A majority of previous simulation methods in computer graphics use 2D and 3D meshes. Most of these approaches are based on mass-spring systems, or the more mathematically motivated Finite Element (FEM), Finite Difference (FD) or Finite Volume (FVM) Methods, in conjunction with elasticity theory. In mesh based approaches, complex physical effects, such as melting, solidifying, splitting or fusion, pose great challenges in terms of restructuring. Additionally, under large deformations the original meshes may become arbitrarily ill-conditioned. For the simulation of these complex physical phenomena, efficient and consistent surface and volume representations are needed, which allow simple restructuring. Our goal is, therefore, to unify the simulation of materials ranging from stiff elastic to highly plastic into one framework, using a mesh free, point-based volume and surface representation, which omits explicit connectivity information and, thus, implicitly encompasses the complex physical effects described above.

In the field of mesh based methods, the trend went from mass-spring systems to approaches based on continuum mechanics: tuning a mass-spring network to get a certain behavior is a difficult task, whereas continuum mechanics parameters can be looked up in text books. The two main mesh free methods that have been employed in computer graphics are particle systems based on Lennard-Jones interaction forces and Smoothed Particle Hydrodynamics (SPH) methods. The former is borrowed from Molecular Dynamics while the latter was designed for the simulation of astrophysical processes. Both methods require parameter tuning as in the case of mass-spring systems. Following the trend in mesh based methods, we propose to take the same steps for mesh free methods and derive forces from elasticity theory.

1.1. Our Contributions

The main contributions of our work to the field of computer graphics are:

a mesh free, continuum mechanics based model for the animation of elastic, plastic and melting objects, and
a dynamically adapting, point-sampled surface for physically based animation.

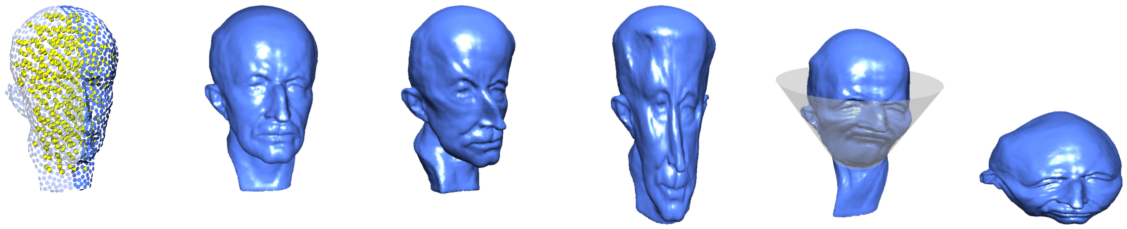


Figure 1: *Left two images: to compute realistic deformations, we represent both the physical volume elements (phyxels) and the surface elements (surfels) as point samples. Right four images: With our system we can model and animate elastic, plastic, melting and solidifying objects.*

Using MLS for the interpolation of point sampled functions is a well known approach in mesh free methods. We introduce it to the computer animation community. However, on top of the MLS based computation of the gradient of the displacement vector field (the tensor field $\nabla \mathbf{u}$), we derive elastic forces in accordance with a linear displacement, constant strain, Finite Element approach. To the best of our knowledge, this combination and the resulting equations are new. In contrast to most standard mesh free approaches, which require solving complex integrals numerically, our method yields simple explicit equations which are easy to code and result in fast simulations.

We can handle high resolution geometry, but also revert to simpler, lower resolution surface modeling and rendering techniques. By introducing this trade-off, our simulation is suitable for both real-time interaction as well as high quality off-line rendering.

1.2. Related Work

An excellent, but somewhat dated survey of deformable modeling is given in [GM97]. In this section we provide an overview of existing work which we found to be most relevant and also related to our own system. A complete survey is beyond the scope of this paper.

1.2.1. Mesh Based Physical Models

Pioneering work in the field of physically-based animation was carried out by Terzopoulos and his co-workers. In their seminal paper [TPBF87] the dynamics of deformable models are computed from the potential energy stored in the elastically deformed body using finite difference (FD) discretization. This work is extended in [TF88, TW88], where the model is extended to cover plasticity and fracture. A hybrid mesh/particle method for heating and melting deformable models is given in [TPF89].

A large number of *mesh based* methods for both off-line and interactive simulation of deformable objects have been proposed in the field of computer graphics. Examples are mass-spring systems used for cloth simulation [BW98, DSB99], the Boundary Element Method (BEM) [JP99] and the Finite Element Method (FEM), which has been employed for the simulation of elastic objects [DDCB01, GKS02], plastic deformation [OBH02] and fracture [OH99].

1.2.2. Mesh Free Physical Models

Our approach to deformable modeling is greatly inspired by so-called *mesh free* or *meshless methods for the solution of partial differential equations*, which, according to [Suk03], originated in the FEM community approximately a decade ago [NTVR92]. An introduction to the element-free Galerkin method is given in [Ask97]. For a more extensive and up-to-date classification and overview of mesh free methods, see [FM03, Liu02, BKO*96].

Desbrun and Cani were among the first to use mesh free models in computer graphics. In [DC95], soft, inelastic substances that can split and merge are animated by combining particle systems with simple inter-particle forces and implicit surfaces for collision detection and rendering. The Smoothed Particle Hydrodynamics (SPH) method [Mon92] is applied in [DC96]: discrete particles are used to compute approximate values of physical quantities and their spatial derivatives. Space-adaptivity is added in [DC99]. In the work of Tonnesen on *particle volumes* [Ton98], elastic inter-particle forces are computed using the Lennard-Jones potential energy function (commonly used to model the interaction potential between pairs of atoms in molecular dynamics).

1.2.3. Point Based Surface Modeling

Our high quality surface representation is based on point samples and draws heavily from existing research. [ST92] and [WH94] proposed the use of point primitives in the context of 3D shape modeling. We apply their idea of sample splitting and merging to dynamically adapt the surface sampling density during simulation. To maintain a close connection between physical particles and surface samples, we use a space warping approach, similar to the free-form shape deformation scheme proposed in [PKKG03]. Our method also

bears some resemblance to projection-based surface models such as [ABCO*03] that implicitly define a continuous surface from an unstructured point cloud. We use a linear version of the Moving Least Squares projection for dynamic surface reconstruction, similar to [AA03].

1.3. Overview

We first give a brief overview of elasticity theory, required to explain our method (Section 2). We then describe our simulation loop in detail in Section 3. An extension of the algorithm to the simulation of plastic and highly deformable substances is developed in Section 4. Section 5 shows two methods for animating the detailed surface along with the physical model. Results are presented in Section 6, after which we conclude with a brief discussion and areas for future work.

2. Elasticity Model

2.1. Continuum Equations

The continuum elasticity equations describe how to compute the elastic stresses inside a volumetric object, based on a given deformation field [Coo95, Chu96]. Consider a model of a 3-dimensional body whose material coordinates are $\mathbf{x} = (x,y,z)^T$. To describe the deformed body, a continuous displacement vector field $\mathbf{u} = (u,v,w)^T$ is used where the scalar displacements $u = u(x,y,z), v = v(x,y,z), w = w(x,y,z)$ are functions of the material coordinates. The coordinates of a point originally located at $\mathbf{x}$ are $\mathbf{x}+\mathbf{u}$ in the deformed model. The Jacobian of this mapping is given by

$$\mathbf{J} = \mathbf{I} + \nabla \mathbf{u}^T = \begin{bmatrix} u_{,x}+1 & u_{,y} & u_{,z} \\ v_{,x} & v_{,y}+1 & v_{,z} \\ w_{,x} & w_{,y} & w_{,z}+1 \end{bmatrix}, \qquad (1)$$

with the following column and row vectors

$$\mathbf{J} = \left[\mathbf{J}_x, \mathbf{J}_y, \mathbf{J}_z\right] = \begin{bmatrix} \mathbf{J}_u^T \\ \mathbf{J}_v^T \\ \mathbf{J}_w^T \end{bmatrix}. \qquad (2)$$

The subscripts with commas represent partial derivatives. To measure strain, we use the quadratic Green-Saint-Venant strain tensor

$$\varepsilon = \mathbf{J}^T\mathbf{J} - \mathbf{I} = \nabla \mathbf{u} + \nabla \mathbf{u}^T + \nabla \mathbf{u} \nabla \mathbf{u}^T. \qquad (3)$$

We assume a Hookean material, meaning that the stress σ linearly depends on the strain ε:

$$\sigma = \mathbf{C}\,\varepsilon, \qquad (4)$$

where $\mathbf{C}$ is a rank four tensor, approximating the constitutive law of the material, and both ε and σ are symmetric 3×3 (rank two) tensors. For an isotropic material, $\mathbf{C}$ has only two independent coefficients, namely Young's Modulus E and Poission's Ratio ν. By modifying $\mathbf{C}$, we can easily incorporate more sophisticated (i.e. anisotropic) constitutive laws into our system. We compute the elastic body forces via the strain energy density:

$$U = \frac{1}{2}(\varepsilon \cdot \sigma) = \frac{1}{2}\left(\sum_{i=1}^{3}\sum_{j=1}^{3} \varepsilon_{ij}\sigma_{ij}\right). \qquad (5)$$

The elastic force per unit volume at a point $\mathbf{x}_i$ is the negative gradient of the strain energy density with respect to this point's displacement $\mathbf{u}_i$ (the *directional derivative* $\nabla_{\mathbf{u}_i}$). For a Hookean material, this expression can be written as

$$\mathbf{f}_i = -\nabla_{\mathbf{u}_i} U = -\frac{1}{2}\nabla_{\mathbf{u}_i}(\varepsilon \cdot \mathbf{C}\varepsilon) = -\sigma \nabla_{\mathbf{u}_i}\varepsilon. \qquad (6)$$

2.2. Volume Conservation

Green's strain tensor given in Eqn. (3) measures linear elongation (normal strain) and alteration of angles (shear strain) but is zero for a volume inverting displacement field. Thus, volume inversion does not cause any restoring elastic body forces. To solve this problem, we add another energy term

$$U_v = \frac{1}{2}k_v(|\mathbf{J}|-1)^2 \qquad (7)$$

that penalizes deviations of the determinant of the Jacobian from positive unity, i.e. deviations from a *right handed* volume conserving transformation. The corresponding body forces are

$$\mathbf{f}_i = -\nabla_{\mathbf{u}_i} U_v = -k_v(|\mathbf{J}|-1)\nabla_{\mathbf{u}_i}|\mathbf{J}|. \qquad (8)$$

2.3. Spatial Discretization

In order to use these continuous elasticity equations in a numerical simulation, the volume needs to be discretized. In mesh based approaches, such as the Finite Element Method (FEM), the volume is divided into elements of finite size. In contrast, in mesh free methods the volume is sampled at a finite number of point locations without connectivity information and without the need of generating a volumetric mesh.

In a mesh free model, all the simulation quantities, such as location $\mathbf{x}_i$, density ρ_i, deformation $\mathbf{u}_i$, velocity $\mathbf{v}_i$, strain ε_i, stress σ_i and body force $\mathbf{f}_i$, are carried by the physically simulated points (actually *point samples*), for which we use the term *phyxel* (physical element) from here on. For each simulated phyxel we have positions $\mathbf{x}_i$ in body space, defining what we call the *reference shape*, and their deformed locations $\mathbf{x}_i + \mathbf{u}_i$ the *deformed shape*.

3. Dynamic Simulation

3.1. Overview

From a high-level view and for each time step Δt, our simulation loop can be described as follows

$$\underbrace{\mathbf{u}_t}_{\text{displacements}} \rightarrow \underbrace{\nabla \mathbf{u}_t}_{\text{derivatives}} \rightarrow \underbrace{\varepsilon_t}_{\text{strains}} \rightarrow \underbrace{\sigma_t}_{\text{stresses}} \rightarrow \underbrace{\mathbf{f}_t}_{\text{forces}} \rightarrow \underbrace{\mathbf{u}_{t+\Delta t}}_{\text{integration}}$$

After initialization (Section 3.2) the simulation loop is started. From the displacement vectors $\mathbf{u}_i$, we approximate the nine spatial derivatives of three scalar fields u, v and w (Section 3.3), which define both the strain and stress tensors (Section 3.4). The forces acting at the points are then computed as the negative gradient of the strain energy with respect to the displacements (Section 3.5) and integrated in time (Section 3.6), yielding new displacements $\mathbf{u}_{t+\Delta t}$ at time $t+\Delta t$.

3.2. Initialization

In continuum mechanics, quantities are measured per unit volume. It is thus important to know how much volume each phyxel represents. We compute the mass m_i, density ρ_i and the volume v_i of a phyxel i as it is done in SPH. Each phyxel has a fixed mass m_i that does not change through the simulation. This mass is distributed around the phyxel via the polynomial kernel

$$W(r,h) = \begin{cases} \frac{315}{64\pi h^9}(h^2 - r^2)^3 & \text{if } r < h \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

proposed in [MCG03], where r is the distance to the phyxel and h is the support of the kernel. This kernel is normalized meaning $\int_{\mathbf{x}} W(|\mathbf{x}-\mathbf{x}_0|,h)d\mathbf{x} = 1$, and has the unit $[1/m^3]$. The density at phyxel i can then be computed as

$$\rho_i = \sum_j m_j w_{ij}, \tag{10}$$

where $w_{ij} = W(|\mathbf{x}_j - \mathbf{x}_i|, h_i)$. Finally, the volume represented by phyxel i is simply given by $v_i = m_i/\rho_i$. While the mass represented by a phyxel is fix, the density and volume vary when the reference positions of the phyxels change in case of plastic deformation (Section 4.2).

The masses m_i and support radii h_i need to be initialized before the simulation starts. We allow irregular initial sampling of the volume. For each phyxel i we compute the average distance $\bar{r}_i$ to its k nearest neighbors (we chose $k = 10$). The support radius h_i is chosen to be a multiple of $\bar{r}_i$ (we chose $h_i = 3\bar{r}_i$). The masses are initialized as $m_i = s\ \bar{r}_i^3\ \rho$, where ρ is the material density and s is the same scaling factor for all phyxels, chosen such that the ρ_i resulting from Eqn. (10) are close to ρ.

3.3. Moving Least Squares Approximation of $\nabla\mathbf{u}$

In order to compute strain, stress and the elastic body forces, the spatial derivatives of the displacement field $\nabla\mathbf{u}$ are needed (see Eqn. (1)). These derivatives are estimated from the displacement vectors $\mathbf{u}_j$ of nearby phyxels. To determine neighboring phyxels, we use spatial hashing [THM*03].

The approximation of $\nabla\mathbf{u}$ must be first order accurate in order to guarantee zero elastic forces for rigid body modes. We therefore compute derivatives using a Moving Least Squares formulation [LS81] with a linear basis. Let us consider the x-component u of the displacement field $\mathbf{u} = (u,v,w)^T$. Using a Taylor approximation, the continuous scalar field $u(\mathbf{x})$ in the neighborhood of $\mathbf{x}_i$ can be approximated as

$$u(\mathbf{x}_i + \Delta\mathbf{x}) = u_i + \nabla u|_{\mathbf{x}_i} \cdot \Delta\mathbf{x} + O(||\Delta\mathbf{x}||^2), \tag{11}$$

where $\nabla u|_{\mathbf{x}_i} = (u_{,x}, u_{,y}, u_{,z})^T$ at phyxel i. Given u_i and the spatial derivatives ∇u at phyxel i, we can approximate the values u_j at close phyxels j as

$$\tilde{u}_j = u_i + \nabla u|_{\mathbf{x}_i} \cdot \mathbf{x}_{ij} = u_i + \mathbf{x}_{ij}^T \nabla u|_{\mathbf{x}_i}, \tag{12}$$

where $\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i$. We measure the error of the approximation as the sum of the squared differences between the approximated values $\tilde{u}_j$ and the known values u_j, weighted by the kernel given in Eqn. (9)

$$e = \sum_j (\tilde{u}_j - u_j)^2\ w_{ij} \tag{13}$$

The differences are weighted because only phyxels in the neighborhood of phyxel i should be considered and, additionally, fade in and out smoothly. Substituting Eqn. (12) into Eqn. (13) and expanding yields $e = \sum_j (u_i + u_{,x}\ x_{ij} + u_{,y}\ y_{ij} + u_{,z}\ z_{ij} - u_j)^2\ w_{ij}$, where x_{ij}, y_{ij} and z_{ij} are the x, y and z-components of $\mathbf{x}_{ij}$ respectively. Given the positions of the phyxels $\mathbf{x}_i$ and the sampled values u_i we want to find the derivatives $u_{,x}$, $u_{,y}$ and $u_{,z}$ that minimize the error e. Setting the derivatives of e with respect to $u_{,x}$, $u_{,y}$ and $u_{,z}$ to zero yields three equations for the three unknowns

$$\left(\sum_j \mathbf{x}_{ij}\mathbf{x}_{ij}^T w_{ij}\right) \nabla u|_{\mathbf{x}_i} = \sum_j (u_j - u_i)\mathbf{x}_{ij} w_{ij} \tag{14}$$

The 3 by 3 system matrix $\mathbf{A} = \sum_j \mathbf{x}_{ij}\mathbf{x}_{ij}^T w_{ij}$ (the moment matrix) can be pre-computed, inverted and used for the computation of the derivative of v and w as well. If $\mathbf{A}$ is non-singular we have the following formula for the computation of derivatives:

$$\nabla u|_{\mathbf{x}_i} = \mathbf{A}^{-1}\left(\sum_j (u_j - u_i)\mathbf{x}_{ij} w_{ij}\right). \tag{15}$$

However, if the number of phyxels within the support radius h in the neighborhood of phyxel i is less than 4 (including phyxel i) or if these phyxels are co-planar or co-linear $\mathbf{A}$ is singular and cannot be inverted. This only happens if the sampling of the volume is too coarse. To avoid problems with singular or badly conditioned moment matrices, we use safe inversion via Singular Value Decomposition [PTVF92].

3.4. Updating Strains and Stresses

With Eqn. (15) we compute for each simulated phyxel i the spatial derivatives of the deformation field at the phyxel's location $\mathbf{x}_i$ based on the displacement vectors $\mathbf{u}_j$ of neighboring phyxels j. Using Eqns. (1), (3) and (4) the Jacobian $\mathbf{J}_i$,

the strain ε_i and the stress σ_i at phyxel i can all be computed from these derivatives:

$$\mathbf{J}_i \leftarrow \begin{bmatrix} \nabla u|_{\mathbf{x}_i}^T \\ \nabla v|_{\mathbf{x}_i}^T \\ \nabla w|_{\mathbf{x}_i}^T \end{bmatrix} + \mathbf{I}, \quad \varepsilon_i \leftarrow (\mathbf{J}_i^T \mathbf{J}_i - \mathbf{I}), \quad \sigma_i \leftarrow (\mathbf{C}\,\varepsilon_i).$$

3.5. Computation of Forces via Strain Energy

As a basic unit, analogous to a finite element in FEM, we consider a phyxel i and all its neighbors j that lie within its support radius h_i (see Fig. 2).

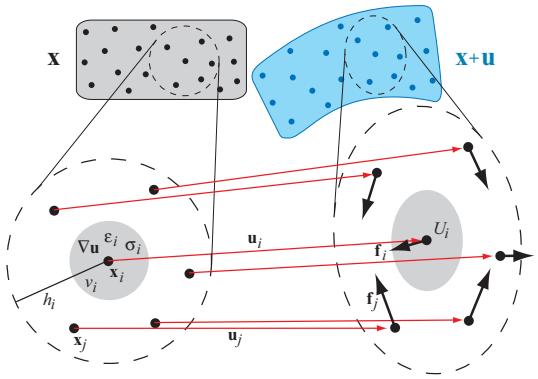


Figure 2: *As a basic unit, we consider a phyxel at $\mathbf{x}_i$ and its neighbors at $\mathbf{x}_j$ within distance h_i. The gradient of the displacement field $\nabla\mathbf{u}$ is computed from the displacement vectors $\mathbf{u}_i$ and $\mathbf{u}_j$, the strain ε_i from $\nabla\mathbf{u}$, the stress σ_i from ε_i, the strain energy U_i from ε_i, σ_i and the volume v_i and the elastic forces as the negative gradient of U_i with respect to the displacement vectors.*

Based on Eqn. (5) we estimate the strain energy stored around phyxel i as

$$U_i = v_i \frac{1}{2}(\varepsilon_i \cdot \sigma_i) \tag{16}$$

assuming that strain and stress are constant within the rest volume v_i of phyxel i, equivalent to using linear shape functions in FEM. The strain energy is a function of the displacement vector $\mathbf{u}_i$ of phyxel i and the displacements $\mathbf{u}_j$ of all its neighbors. Taking the derivative with respect to these displacements using Eqn. (6) yields the forces acting at phyxel i and all its neighbors j

$$\mathbf{f}_j = -\nabla_{\mathbf{u}_j} U_i = -v_i \sigma_i \nabla_{\mathbf{u}_j} \varepsilon_i \tag{17}$$

The force acting on phyxel i turns out to be the negative sum of all $\mathbf{f}_j$ acting on its neighbors j. These forces conserve linear and angular momentum.

Using Eqn. (15), this result can be further simplified (see Appendix A) to the compact form

$$\mathbf{f}_i = -2v_i \mathbf{J}_i \sigma_i \mathbf{d}_i = \mathbf{F}_e \mathbf{d}_i \tag{18}$$

$$\mathbf{f}_j = -2v_i \mathbf{J}_i \sigma_i \mathbf{d}_j = \mathbf{F}_e \mathbf{d}_j, \tag{19}$$

where

$$\mathbf{d}_i = \mathbf{A}^{-1}\left(-\sum_j \mathbf{x}_{ij} w_{ij}\right) \tag{20}$$

$$\mathbf{d}_j = \mathbf{A}^{-1}(\mathbf{x}_{ij} w_{ij}) \tag{21}$$

For the volume conserving force defined in Eqn. (8) using Eqn. (15) we get

$$\mathbf{f}_i = -v_i k_v (|\mathbf{J}| - 1) \begin{bmatrix} (\mathbf{J}_v \times \mathbf{J}_w)^T \\ (\mathbf{J}_w \times \mathbf{J}_u)^T \\ (\mathbf{J}_u \times \mathbf{J}_v)^T \end{bmatrix} \mathbf{d}_i = \mathbf{F}_v \mathbf{d}_i \tag{22}$$

$$\mathbf{f}_j = \mathbf{F}_v \mathbf{d}_j \tag{23}$$

Using the definition of the vectors $\mathbf{d}_i$ and $\mathbf{d}_j$ we get for the total internal forces:

$$\mathbf{f}_i = (\mathbf{F}_e + \mathbf{F}_v)\,\mathbf{A}^{-1}\left(-\sum_j \mathbf{x}_{ij} w_{ij}\right) \tag{24}$$

$$\mathbf{f}_j = (\mathbf{F}_e + \mathbf{F}_v)\,\mathbf{A}^{-1}(\mathbf{x}_{ij} w_{ij}) \tag{25}$$

The matrix product $(\mathbf{F}_e + \mathbf{F}_v)\,\mathbf{A}^{-1}$ is independent of the individual neighbor j and needs to be computed only once for each phyxel i in each time step Δt.

3.6. Time Integration

The elastic strain energy U_i of a phyxel defined in Eqn. (16) is an energy, not an energy density, because we multiply by the phyxel's rest volume v_i. Thus, the elastic force derived from it is a force and not a force per unit volume. The acceleration $\mathbf{a}_i$ of a phyxel due to this force is therefore

$$\frac{d^2\mathbf{u}_i}{dt^2} = \mathbf{a}_i = \mathbf{f}_i / m_i. \tag{26}$$

A large number of time integration schemes have been proposed. Explicit schemes are easy to implement and computationally cheap, but stable only for small time steps. In contrast, implicit schemes are unconditionally stable but, computationally and in terms of memory consumption, more complex. We found that for our simulations, a simple Leap-Frog scheme performs best in this trade-off. However, we also provide the tangent stiffness matrix derived from the elastic forces for implicit integration in Appendix B.

4. Plasticity Model

4.1. Strain State Plasticity

An elegant way of simulating plastic behavior, is by using strain state variables [OBH02]. Every phyxel i stores a plastic strain tensor ε_i^p. The strain considered for elastic forces $\tilde{\varepsilon}_i = \varepsilon_i - \varepsilon_i^p$ is the difference between measured strain ε_i and the plastic strain. Thus, in case the measured strain equals the plastic strain, no forces are generated. Since ε_i^p is considered constant within one time step, the elasto-plastic forces

(Equations (18) and (19)) are computed using $\tilde{\sigma}_i = \mathbf{C}\,\tilde{\varepsilon}_i$ instead of σ_i. The plastic strain is updated at every time step according to [OBH02].

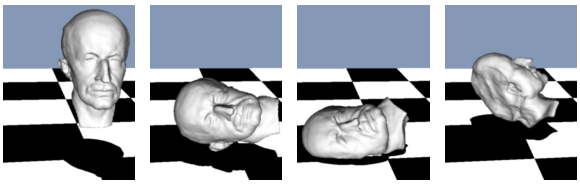


Figure 3: *An animation sequence, in which we elastically and plastically deform Max Planck in real-time, switching between material properties on the fly.*

4.2. Deformation of Reference Shape

In contrast to mesh based methods, the mesh free approach is particulary useful when the object deviates far from its original shape in which case the original mesh connectivity is not useful anymore. Using a mesh free method, the reference shape can easily adapt to the deformed shape. However, changing the reference positions of phyxels is dangerous: two phyxels from two different objects having reference positions $\mathbf{x}_i$ and $\mathbf{x}_j$ might move within each others support, even though their actual positions $\mathbf{x}_i + \mathbf{u}_i$ and $\mathbf{x}_j + \mathbf{u}_j$ are far from each other. This large displacement vector difference results in large strains, stresses and elastic forces, causing the simulation to crash. Therefore, if the reference shape is changed, both reference shape and deformed shape need to be kept close to each other. We have found a very simple way to achieve this, with which we can model highly plastic materials, melting and flow: after each time step, we absorb the deformation completely while storing the built up strains in the plastic strain state variable.

forall phyxels i **do**
  $\varepsilon_i^p \leftarrow \varepsilon_i^p - \varepsilon_i$
  $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{u}_i$
  $\mathbf{u}_i \leftarrow \mathbf{0}$
endfor
forall phyxels i **do** update ρ_i, v_i and $\mathbf{A}_i$ **endfor**

This way, both reference shape and deformed shape are identical after each time step. The strain is not lost, but stored in the plastic state variable. As our results show, this procedure generates appealing animations of highly deformable and plastic materials (Fig. 5).

The volume conserving term described in Section 2.2 measures volume deviations from the reference shape to the deformed shape. Because the original shape information is lost, the term cannot guarantee volume conservation over time. Therefore, for melting and flow experiments, we add pressure forces based on SPH as suggested in [DC99] and use their spiky kernel [DC96] for density computations.

5. Surface Animation

In this section, we describe the animation of the point-sampled surface after a simulation step. First, we present a fast approach which makes use of the continuously defined displacement vector field $\mathbf{u}(x)$. We then extend this approach to a high quality surface animation algorithm, which can handle arbitrary topological changes while still preserving detail by employing multiple surface representations. In the following, the term *surface elements* (or *surfels* for short) denotes the point-sampled surface representation.

5.1. Displacement Approach

The idea of the displacement-based approach is to carry the surfels along with the phyxels. The displacement vector $\mathbf{u}_{sfl}$ at a known surfel position $\mathbf{x}_{sfl}$ is computed from the displacements $\mathbf{u}_i$ of the neighboring phyxels. For this we need to define a smooth displacement vector field in $\mathbb{R}^3$, which is invariant under linear transformations. This can be achieved by using a first order MLS approximation. However, we have already obtained such an approximation of $\nabla\mathbf{u}$, described in Section 3.3, which we reuse. Thus, we can compute the displacement vector $\mathbf{u}_{sfl}$ as

$$\mathbf{u}_{sfl} = \frac{1}{\sum_i \omega(r_i)} \sum_i \omega(r_i)(\mathbf{u}_i + \nabla\mathbf{u}_i^T(\mathbf{x}_{sfl} - \mathbf{x}_i)), \qquad (27)$$

where $\omega(r_i) = \omega(||\mathbf{x}_{sfl} - \mathbf{x}_i||) = e^{-r_i^2/h^2}$ is a Gaussian weighting function. The $\mathbf{u}_i$ are the displacement vectors of phyxels at $\mathbf{x}_i$ within a distance h to $\mathbf{x}_{sfl}$.

We proceed similar to [PKKG03] by applying the displacement $\mathbf{u}_{sfl}$ to both the surfel center and its tangent axes. A surfel splitting and merging scheme is applied to maintain a high surface quality in the case of large deformations, see [PKKG03] for details.

5.2. Multi-Representation Approach

The multi-representation approach is motivated by two desirable effects: (a) when parts of a model fracture or merge, topological changes need to be handled, and (b) when colliding a highly deformable model with a rigid object (e.g. a casting mold), the model must adapt to the possibly very detailed shape and retain it after solidifying. An example is shown in Fig. 9 where the Max Planck Bust is melted, flows into the Igea model and solidifies. While implicit surfaces can easily represent highly complex topology and guarantee global consistency by construction, explicit representations are more suitable for detailed surfaces and fast rendering. To achieve the effects described above, we employ both representations.

5.2.1. Implicit Representation

Desbrun and Cani suggested using an implicit representation for coating a set of skeletons [DC95]. We use the same idea

to associate a field function $f_i : \mathbb{R}^3 \mapsto \mathbb{R}$ to each phyxel p_i. To obtain a continuously defined field in space, the contributions of all phyxels within a certain distance d to $\mathbf{x}$ are added up, i.e. $F(\mathbf{x}) = \sum_{i,r<d} f_i(\mathbf{x})$. L_I is then defined as an iso-value S of F, i.e. $L_I = \{\mathbf{x} \in \mathbb{R}^3 \mid F(\mathbf{x}) - S = 0\}$.

Many different field functions have been presented in the literature (e.g. [BS95, WW89]). We chose the field function from [Bli82], which is suitable for our application:

$$f_i(\mathbf{x}) = be^{-ar^2}, \tag{28}$$

where $\mathbf{x}$ is an arbitrary point in space and $r = \|\mathbf{x} - \mathbf{x}_i\|$. The constants a and b can be computed as $a = \frac{-B}{R^2}$ and $b = Se^{-B}$, where R is the radius in isolation and B the (negative) blobbiness.

For an initial sampling of the implicit surface, all surfels of the model are projected onto it, i.e. we need to find the projected position $\mathbf{x}_L$ of a surfel $\mathbf{x}$ such that $F(\mathbf{x}_L) - S = 0$. This is a classical root finding problem, we refer to [PTVF92] for further information. We then apply our resampling operator described in Section 5.2.4 to ensure that the implicit surface is hole-free and regularly sampled.

After each animation step, the surfels are animated together with the implicit surface by employing the displacement approach (Section 5.1) to get an estimation of their new position, followed by a projection onto the implicit surface. Finally, the resampling operator is applied again.

5.2.2. Detail Representation

While the implicit surface L_I can easily handle any topological changes, it can represent only blobby surfaces. Therefore we introduce the detail representation L_D which represents a highly detailed model. At the beginning of the animation, L_D is equal to the model surface. When topological changes occur, the surface locally looses its detail and L_D converges locally to L_I.

First we need a metric which quantifies (local) topological change. We refer to the phyxels which lie on the boundary Γ of the phyxel cloud as outside phyxels, and phyxels which are enclosed by the boundary as inside phyxels. We observe, that when a model fractures, inside phyxels will become outside phyxels, and vice-versa if parts are merged, outside phyxels will become inside phyxels. We can determine the probability that a phyxel p belongs to Γ by Principal Component Analysis of the neighboring phyxels. The idea is, that if we look at a phyxel inside the volume, all eigenvalues are expected to be similar, while for a phyxel on the boundary, one eigenvalue is expected to be small. We can efficiently compare the eigenvalues by computing the trace and the determinant of the local covariance matrix $\mathbf{C} = \sum_{i=1}^{n} (\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T$, where $\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$ and n is the number of phyxels in the support radius of p. The determinant $det(\mathbf{C})$ is equal to the product of the three eigenvalues and the trace is equal to their sum. Thus, we can estimate the probability P that p is an outside phyxel as

$$P = 1 - \frac{det(\mathbf{C})}{(trace(\mathbf{C})/3)^3}. \tag{29}$$

We can now estimate the local change of topology $\Delta\Gamma^t$ at time step t by comparing the sum P_i of the neighboring phyxels of a surfel with the last time step, i.e.

$$\Delta\Gamma^t = \frac{\lambda}{n} \left| \sum_{i=1}^{n} P_i^{t-1} - \sum_{i=1}^{n} P_i^t \right|, \tag{30}$$

where λ is a constant weighting factor.

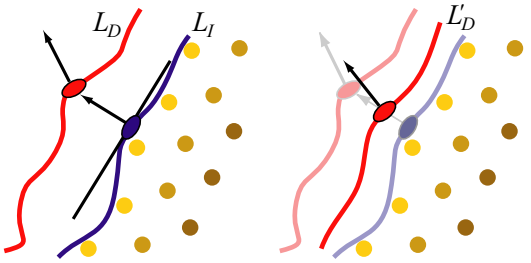


Figure 4: *Left: every surfel from L_D (red) is projected onto L_I (blue). Right: the blending factor between these two positions (and normals) is computed from $\Delta\Gamma$, the estimated physical boundary variation between two time steps. This results in the updated L'_D. Note: lighter phyxels have a higher boundary probability P.*

To transfer L_D to L_I, the function-values $F(\mathbf{x}_{sfl})$ of the surfels need to approach the iso-value S of L_I. Assume a surfel initially has a function-value s^0, its function-value at time step $t-1$ is s^{t-1}, and $\Delta\Gamma^t_{tot} = min(1, \sum_{i=1\ldots t} \Delta\Gamma^i)$. When we assume that the function-values change linearly, we can estimate the new function-value by linear interpolation, i.e. $\tilde{s}^t = s^0 + \Delta\Gamma^t_{tot}(S - s^0)$. We can now use this estimation to compute a blending factor b_I for blending both position and normal of the surfel with its projection onto L_I: $b_I = (\tilde{s}^t - s^{t-1})/(S - s^{t-1})$. The blended surfel positions are computed as $\mathbf{x}'_D = b_I\mathbf{x}_I + (1 - b_I)\mathbf{x}_D$, the normals are blended similarly. After the blending, all surfels in L_D are updated, resulting in L'_D (Fig. 4), thereby discarding the original shape.

5.2.3. Contact Representation

Assume that an object is melted and flows into another object (i.e. a mold), which is itself point-sampled (Fig. 9). In this case, the surfels which change from inside the mold to outside (i.e. collide with the mold) need to adapt to its surface. Whether a surfel is inside or outside can be determined efficiently as described in [PKKG03]. A colliding surfel is then projected onto the MLS surface [Lev01, ABCO*03] of the mold, setting its normals equal to the normal of the MLS surface and interpolating its color from the neighboring surfels.

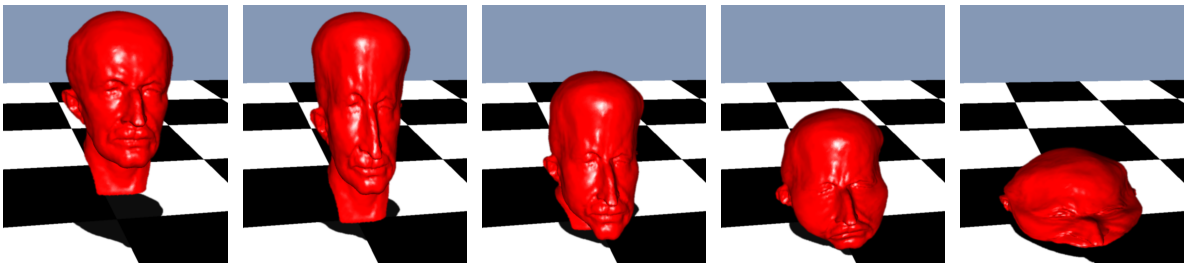


Figure 5: *A soft elastic Max Planck model is initially held by the cranium, then melted and dropped to the ground plane.*

The affected surfels are stored as the *contact representation* L_C and used for rendering instead of the L_D representation.

5.2.4. Resampling

In the case of topological changes, the splitting and merging of surfels as described in Section 5.1 is not sufficient. Because with point-sampled surfaces we do not have to deal with consistency constraints, we can easily resample the surface by iteratively applying a combination of resampling and relaxation. First, we efficiently detect under- and oversampling locally by computing, for each surfel, the number k of nearest neighbors which are in a certain distance to the surfel. This number is compared with two user defined thresholds min_{nb} and max_{nb}. If k is larger than max_{nb}, the surfel is deleted with a probability of $k/(2max_{nb})$. If k is smaller than min_{nb}, the surfel is split $min_{nb} - k$ times and the new surfels are distributed in the tangent plane of the surfel. Our resampling operator requires an approximately uniform distribution of the surfels. To achieve this and to spread the surfels over uncovered surface areas in the case of topological changes, a repulsion scheme is applied to the surfels, similar to [PGK02]. The idea to use a combination of spreading and splitting of the surfels is similar to the particle distribution scheme proposed in [WH94].

5.2.5. Zombies

After a resampling step, the surfels need to be reprojected onto the surface. To avoid error accumulation, we carry along two surface representations: The original surfels (called zombies) and the resampled surfels, whereby only the resampled surfels are displayed. Initially, at the beginning, both representations are equal. When the animation starts, the zombies are animated as described above, but without resampling. The resampled surfels are simply displaced and then projected onto the MLS surface [Lev98, Lev01] defined by the zombie surfels. This works well as long as the topology does not change. Because of the blending, the closest zombie surfel is expected to lie on L_I in this case. Therefore, we check if the function-value of the closest zombie is equal to the iso-surface S. If this is the case we project the surfel onto L_I instead of the MLS surface. Finally, we use the zombie surfels to interpolate attributes like color of the resampled surfels, similar to [PKKG03].

6. Results

6.1. Real-Time Deformation

For real-time demonstrations of our algorithm, we use the displacement-based surface animation approach described in Section 5.1, as it is capable of animating a surface model with 10k surfels and approximately 200 phyxels at 27 frames per second on a P4 2,8 GHz Laptop with an NVidia GeForce FX Go5600 GPU. In Fig. 3 we let an elastic model with $E = 0,5 \cdot 10^6 N/m^2$ bounce off the ground plane: the model exhibits realistic elastic behavior. Shortly before hitting the ground a second time, we switch to a plastic material, resulting in an irreversible dent. Afterwards we switch back to a stiff elastic material with $E = 0,5 \cdot 10^7 N/m^2$. The model has taken considerable damage, but the surface is still skinned correctly. A real-time melting animation with an adaptively sampled surface is shown in Fig. 5, where the model exhibits realistic elastic, plastic, melting and flowing effects.

	physics	surface + rendering	frame rate
Max/200/10k/expl	15 ms	22 ms	27 fps
Max/200/10k/impl	22 ms	22 ms	22 fps
Max/400/20k/expl	35 ms	50 ms	12 fps
Max/400/20k/impl	60 ms	50 ms	9 fps

Table 1: *Timings of our system, running on a 2,8 GHz Pentium 4 Laptop with a GeForce FX Go5600 GPU.*

Some timings of our algorithm are given in Table 1. The first column describes the model, the number of phyxels, the number of surfels and whether explicit or implicit integration was used (model/phyxels/surfels/{expl|impl}).

6.2. High-Quality Surface Animation

To demonstrate the multi-representation approach of Section 5.2, we let a highly plastic 53k surfel model of Max Planck flow through a funnel into the Igea model (134k surfels), as shown in Fig. 9. The Max Planck model is sampled with 600 phyxels. While the Max Planck model is squeezed through the funnel, its detail vanishes (i.e. the detail level approaches the implicit representation L_I). When the surface of the Max Planck model gets in contact with the Igea model, it adapts to detailed features. After the Max Planck model has filled the Igea model, we let the initially highly deformable substance solidify, i.e. we set L_C equal to L_D and reduce the plastic creep. Finally, we let it drop to the ground plane, where it exhibits realistic elastic behavior (see our video). On average, the animation with our high quality software renderer takes 8 seconds per frame on a 2,8 GHz Pentium 4. Due to resampling, the final solidified model consists of 115k surfels.

A second example of our multi-representation approach is shown in Fig. 10. Here, we initially fix the shock of hair of the Igea model. Due to gravitation, part of the model splits off and drops to the plane. Note that at places where no topological changes take place, detail is preserved. Afterwards, we release the shock of hair, letting it merge with the rest of the model. Note that the texture of the model is preserved due to the interpolation of zombies. Initially, the Igea model has 134k surfels. This number is increases to 340k surfels during the animation due to topological changes. After the two parts have merged towards the end of the animation sequence, the number of surfels is 105k.

6.3. Range of Physical Parameters

In Figure 6 we demonstrate the effect of Poisson's Ratio ν for volume conservation. For the image in the middle we set ν to zero. When the model is pulled vertically, its width does not change and its volume is not conserved. In contrast, with a ratio of 0.49 the width adjusts to the stretch thereby conserving the volume of the object (right).

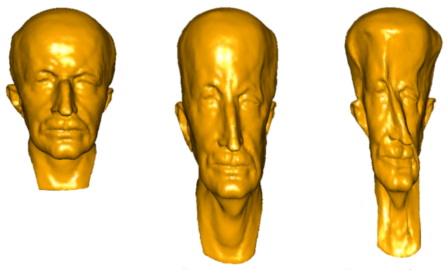


Figure 6: *The effect of Poisson's Ratio: the undeformed model (left) is stretched using a Poisson Ratio of zero (middle) and* 0.49 *(right).*

Figure 7 shows a melting experiment that demonstrates the range of material properties that can be simulated with our method. We first drop a soft cuboid object on a spike. It melts and deforms under gravity with a Young's Modulus of $E = 10^4 N/m^2$, high c_{creep} and $c_{\text{min}} = 0$ (see Section 4.1). The reference shape is deformed along with the deformed shape (see Section 4.2). After the material has come to rest, the Young's Modulus is increased to $E = 10^5 N/m^2$ and c_{creep} is set to zero. The user can then lift the donut up as an elastic object.

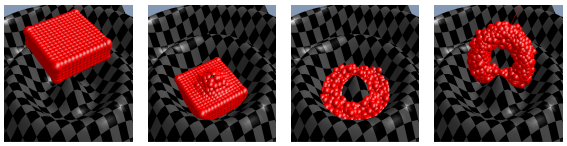


Figure 7: *Demonstration of a change in topology, using the procedure described in Section 4.2. A highly plastic material melts into a circular well. After the material is made elastic, the user can lift up the resulting donut.*

Using implicit integration (with a time step of 0.01 seconds), we can set the Young's Modulus E as high as $10^8 N/m^2$ without stability problems. When setting E to $10^6 N/m^2$, explicit integration (with 15-20 time steps of 0.0001 seconds per animation frame) also works stable at interactive rates. Setting $E = 10^5 N/m^2$ yields very soft objects, which exhibit detailed, local deformations under external excitation. These sometimes unwanted oscillating effects are damped out using implicit integration.

7. Limitations

In its current state, our model has a few limitations.

- We assume a Hookean material. Therefore, the model allows material anisotropy, but only linear stress-strain relationships.
- MLS (Section 3.3) only works well if each phyxel has at least three neighbors at non-degenerate locations, thus the approach only works for volumes, not for 2D layers or 1D strings of phyxels.
- Close phyxels always interact, so for fracture simulations, the model would have to be extended. Furthermore, the current surface animation algorithm supports fractures to a limited extent only. In particular, we will have to extend it to cope with sharp features, which occur in brittle material fracturing.

8. Conclusions and Future Work

In this paper we have presented a novel, mesh-free animation algorithm derived from continuum mechanics, which uses point samples for both volume and surface modeling. Our system is capable of simulating a wide range of elastically and plastically deformable objects which can melt, flow, solidify split and merge. An interesting feature is the capability

to switch between any of these physical conditions at runtime, resulting in visually plausible and interesting effects. We have also described methods by which we can animate a detailed surface along with the physical representation, introducing a trade-off between real-time performance and visual fidelity.

In general, the engineering field of mesh free methods is a vast, yet relatively new area, in which we see great potential for computer graphics, animation and simulation research.

References

[AA03] ADAMSON A., ALEXA M.: Approximating and intersecting surfaces from points. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2003), pp. 230–239.

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Computing and rendering point set surfaces. *IEEE TVCG 9*, 1 (2003), 3–15.

[Ask97] ASKES H.: *Everything you always wanted to know about the Element-Free Galerkin method, and more*. Tech. rep., TU Delft nr. 03.21.1.31.29, 1997.

[BKO*96] BELYTSCHKO T., KRONGAUZ Y., ORGAN D., FLEMING M., KRYSL P.: Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering 139*, 3 (1996), 3–47.

[Bli82] BLINN J. F.: A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3 (1982), 235–256.

[BS95] BLAC C., SCHLICK C.: Extended field functions for soft objects. In *Implicit Surfaces'95* (1995), pp. 21–32.

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of SIGGRAPH 1998* (1998), pp. 43–54.

[Chu96] CHUNG T. J.: *Applied Continuum Mechanics*. Cambridge Univ. Press, NY, 1996.

[Coo95] COOK R. D.: *Finite Element Modeling for Stress Analysis*. John Wiley & Sons, NY, 1995.

[DC95] DESBRUN M., CANI M.-P.: Animating soft substances with implicit surfaces. In *Computer Graphics Proceedings* (1995), ACM SIGGRAPH, pp. 287–290.

[DC96] DESBRUN M., CANI M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *6th Eurographics Workshop on Computer Animation and Simulation '96* (1996), pp. 61–76.

[DC99] DESBRUN M., CANI M.-P.: *Space-Time Adaptive Simulation of Highly Deformable Substances*. Tech. rep., INRIA Nr. 3829, 1999.

[DDCB01] DEBUNNE G., DESBRUN M., CANI M.-P., BARR A.: Dynamic real-time deformations using space & time adaptive sampling. In *Computer Graphics Proceedings* (Aug. 2001), Annual Conference Series, ACM SIGGRAPH 2001, pp. 31–36.

[DSB99] DESBRUN M., SCHRÖDER P., BARR A. H.: Interactive animation of structured deformable objects. In *Graphics Interface '99* (1999).

[FM03] FRIES T.-P., MATTHIES H. G.: *Classification and Overview of Meshfree Methods*. Tech. rep., TU Brunswick, Germany Nr. 2003-03, 2003.

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: CHARMS: A simple framework for adaptive simulation. In *Proceedings of SIGGRAPH 2002* (2002), pp. 281–290.

[GM97] GIBSON S. F., MIRTICH B.: *A survey of deformable models in computer graphics*. Technical Report TR-97-19, MERL, Cambridge, MA, 1997.

[JP99] JAMES D. L., PAI D. K.: Artdefo, accurate real time deformable objects. In *Computer Graphics Proceedings* (Aug. 1999), Annual Conference Series, ACM SIGGRAPH 99, pp. 65–72.

[Lev98] LEVIN D.: The approximation power of moving least-squares. *Math. Comp. 67*, 224 (1998), 1517–1531.

[Lev01] LEVIN D.: Mesh-independent surface interpolation. In *Advances in Computational Mathematics* (2001).

[Liu02] LIU G. R.: *Mesh-Free Methods*. CRC Press, 2002.

[LS81] LANCASTER P., SALKAUSKAS K.: Surfaces generated by moving least squares methods. *Mathematics of Computation 87* (1981), 141–158.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. *Proceedings of 2003 ACM SIGGRAPH Symposium on Computer Animation* (2003), 154–159.

[Mon92] MONAGHAN J.: Smoothed particle hydrodynamics. *Annu. Rev. Astron. and Astrophysics 30* (1992), 543.

[NTVR92] NAYROLES B., TOUZOT G., VILLON P., RICARD A.: Generalizing the finite element method: diffuse approximation and diffuse elements. *Computational Mechanics 10*, 5 (1992), 307–318.

[OBH02] O'BRIEN J. F., BARGTEIL A. W., HODGINS J. K.: Graphical modeling and animation of ductile fracture. In *Proceedings of SIGGRAPH 2002* (2002), pp. 291–294.

[OH99] O'BRIEN J. F., HODGINS J. K.: Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH 1999* (1999), pp. 287–296.

[PGK02] PAULY M., GROSS M., KOBBELT L. P.: Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization '02* (2002), IEEE Computer Society, pp. 163–170.

[PKKG03] PAULY M., KEISER R., KOBBELT L. P., GROSS M.: Shape modeling with point-sampled geometry. *ACM Trans. Graph. 22*, 3 (2003), 641–650.

[PTVF92] PRESS W. H., TEUKOLSKY S. A., VETTERLING W. T., FLANNERY B. P.: *Numerical Recipes in C: The Art of Scientific Computing*, second ed. Cambridge University Press, 1992.

[ST92] SZELISKI R., TONNESEN D.: Surface modeling with oriented particle systems. *Computer Graphics 26*, 2 (1992), 185–194.

[Suk03] SUKUMAR N.: Meshless methods and partition of unity finite elements. In *of the Sixth International ESAFORM Conference on Material Forming* (2003), pp. 603–606.

[TF88] TERZOPOULOS D., FLEISCHER K.: Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (1988), ACM Press, pp. 269–278.

[THM*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANERTS D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. In *Proc. Vision, Modeling, Visualization VMV* (2003), pp. 47–54.

[Ton98] TONNESEN D.: *Dynamically Coupled Particle Systems for Geometric Modeling, Reconstruction, and Animation*. PhD thesis, University of Toronto, November 1998.

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Computer Graphics Proceedings* (July 1987), Annual Conference Series, ACM SIGGRAPH 87, pp. 205–214.

[TPF89] TERZOPOULOS D., PLATT J., FLEISCHER K.: Heating and melting deformable models (from goop to glop). In *Graphics Interface '89* (1989), pp. 219–226.

[TW88] TERZOPOULOS D., WITKIN A.: Physically based models with rigid and deformable components. *IEEE Computer Graphics and Applications 8*, 6 (1988), 41–51.

[WH94] WITKIN A. P., HECKBERT P. S.: Using particles to sample and control implicit surfaces. In *Computer Graphics Proceedings* (1994), ACM SIGGRAPH, pp. 269–277.

[WW89] WYVILL B., WYVILL G.: Field functions for implicit surfaces. In *Visual Computer* (1989).

Appendix A: Derivation of the Elastic Force

According to Eqn. (15) we have for the derivatives of the x-component u of the displacement field:

$$\nabla u = \begin{pmatrix} u_{,x} \\ u_{,y} \\ u_{,z} \end{pmatrix} = \mathbf{A}^{-1} \left(\sum_j (u_j - u_i)\mathbf{x}_{ij} w_{ij} \right) \qquad (31)$$

Taking the derivatives with respect to the displacement $\mathbf{u}_j = (u_j, v_j, w_j)$ of a phyxel other than the center phyxel i yields

$$\frac{\partial}{\partial u_j}\nabla u = \frac{\partial}{\partial u_j}\mathbf{J}_u = \mathbf{A}^{-1}\left(\mathbf{x}_{ij}w_{ij}\right) = \mathbf{d}_j \quad (32)$$

$$\frac{\partial}{\partial v_j}\nabla u = \frac{\partial}{\partial v_j}\mathbf{J}_u = \mathbf{0} \quad (33)$$

$$\frac{\partial}{\partial w_j}\nabla u = \frac{\partial}{\partial w_j}\mathbf{J}_u = \mathbf{0} \quad (34)$$

with analog results for ∇v and ∇w. For the center phyxel i the vector $\mathbf{d}_j$ needs to be replaced by the vector $\mathbf{d}_i = -\sum_j \mathbf{d}_j$.

Green's strain tensor is defined in Eqn. (3). For the six independent components of the strain tensor we, thus, have

$$\begin{bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \varepsilon_{zz}\\ \varepsilon_{xy}\\ \varepsilon_{yz}\\ \varepsilon_{zx}\end{bmatrix} = \begin{bmatrix} 2u_{,x}+u_{,x}u_{,x}+v_{,x}v_{,x}+w_{,x}w_{,x}\\ 2v_{,y}+u_{,y}u_{,y}+v_{,y}v_{,y}+w_{,y}w_{,y}\\ 2w_{,z}+u_{,z}u_{,z}+v_{,z}v_{,z}+w_{,z}w_{,z}\\ u_{,y}+v_{,x}+u_{,x}u_{,y}+v_{,x}v_{,y}+w_{,x}w_{,y}\\ v_{,z}+w_{,y}+u_{,y}u_{,z}+v_{,y}v_{,z}+w_{,y}w_{,z}\\ w_{,x}+u_{,z}+u_{,z}u_{,x}+v_{,z}v_{,x}+w_{,z}w_{,x}\end{bmatrix}. \quad (35)$$

Taking the derivative of the first strain component ε_{xx} with respect to $\mathbf{u}_j$ using Eqns. (32) - (34) yields

$$\nabla_{\mathbf{u}_j}\varepsilon_{xx} = 2\begin{bmatrix}\mathbf{J}_x & \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{d}_j \quad (36)$$

The derivatives of the remaining strain components are

$$\nabla_{\mathbf{u}_j}\varepsilon_{yy} = 2\begin{bmatrix}\mathbf{0} & \mathbf{J}_y & \mathbf{0}\end{bmatrix}\mathbf{d}_j \quad (37)$$

$$\nabla_{\mathbf{u}_j}\varepsilon_{zz} = 2\begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{J}_z\end{bmatrix}\mathbf{d}_j \quad (38)$$

$$\nabla_{\mathbf{u}_j}\varepsilon_{xy} = \begin{bmatrix}\mathbf{J}_y & \mathbf{J}_x & \mathbf{0}\end{bmatrix}\mathbf{d}_j \quad (39)$$

$$\nabla_{\mathbf{u}_j}\varepsilon_{yz} = \begin{bmatrix}\mathbf{0} & \mathbf{J}_z & \mathbf{J}_y\end{bmatrix}\mathbf{d}_j \quad (40)$$

$$\nabla_{\mathbf{u}_j}\varepsilon_{zx} = \begin{bmatrix}\mathbf{J}_z & \mathbf{0} & \mathbf{J}_x\end{bmatrix}\mathbf{d}_j \quad (41)$$

Finally, according to Eqn. (6) for the body force at phyxel j we get

$$\begin{aligned}\mathbf{f}_j = & -v_i(\sigma_{xx}\nabla_{\mathbf{u}_j}\varepsilon_{xx}+\sigma_{yy}\nabla_{\mathbf{u}_j}\varepsilon_{yy}+\sigma_{zz}\nabla_{\mathbf{u}_j}\varepsilon_{zz}\\ & +2\sigma_{xy}\nabla_{\mathbf{u}_j}\varepsilon_{xy}+2\sigma_{yz}\nabla_{\mathbf{u}_j}\varepsilon_{yz}+2\sigma_{zx}\nabla_{\mathbf{u}_j}\varepsilon_{zx})\\ = & \quad 2v_i\mathbf{J}\sigma\,\mathbf{d}_j\end{aligned}$$

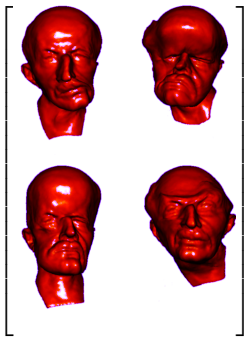
$\mathbf{M}_{maxplanck} =$ 

Figure 8: *A 2×2 "matrix" of deformed Max Plancks.*

Appendix B: Implicit Integration

Using implicit Euler integration, the positions and velocities of all phyxels i at the next time step are computed as follows

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t\mathbf{v}^{t+1} \quad (42)$$

$$\mathbf{M}\mathbf{v}^{t+1} = \mathbf{M}\mathbf{v}^t + \Delta t F(\mathbf{x}^{t+1}), \quad (43)$$

where the vectors $\mathbf{x}$ and $\mathbf{v}$ contain the positions and velocities respectively of all phyxels in the system. The matrix $\mathbf{M}$ is diagonal and contains the masses of the phyxels on its diagonal. The function F computes all body forces based on the positions of all phyxels. To compute the velocities at the next time step, Eqn. (42) is substituted into Eqn. (43) and F is approximated linearly:

$$\begin{aligned}\mathbf{M}\mathbf{v}^{t+1} = & \quad \mathbf{M}\mathbf{v}^t + \Delta t F(\mathbf{x}^t + \Delta t\mathbf{v}^{t+1}) \quad (44)\\ \approx & \quad \mathbf{M}\mathbf{v}^t + \Delta t F(\mathbf{x}^t) + \Delta t^2\mathbf{K}|_{\mathbf{x}^t}\cdot\mathbf{v}^{t+1}, \quad (45)\end{aligned}$$

where $\mathbf{K}|_{\mathbf{x}^t} = \nabla_{\mathbf{x}}F(\mathbf{x}^t)$ is the Jacobian of F and the *tangent stiffness matrix* of the system evaluated at position $\mathbf{x}^t$. By rearranging the above equation, we get the linear system for the unknown velocities $\mathbf{v}^{t+1}$

$$\left(\mathbf{M} - \Delta t^2\mathbf{K}|_{\mathbf{x}^t}\right)\mathbf{v}^{t+1} = \mathbf{M}\mathbf{v}^t + \Delta t F(\mathbf{x}^t) \quad (46)$$

Once the new velocities $\mathbf{v}^{t+1}$ are known, Eqn. (42) can be used to compute the new positions $\mathbf{x}^{t+1}$ explicitly. We now derive the stiffness matrix $\mathbf{K}|_{\mathbf{x}} = \nabla_{\mathbf{x}}F(\mathbf{x})$ for the elastic forces described in Section 3.5. For n phyxels $\mathbf{K}$ is $3n\times 3n$ dimensional and composed of 3×3 dimensional blocks $\mathbf{K}_{kl}|\, k,l \in (1\ldots n)$. The submatrix $\mathbf{K}_{kl}$ has the form

$$\mathbf{K}_{kl} = \nabla_{\mathbf{u}_l}\mathbf{f}_k = \left(\frac{d}{du_l}\mathbf{f}_k, \frac{d}{dv_l}\mathbf{f}_k, \frac{d}{dw_l}\mathbf{f}_k\right) \quad (47)$$

and describes the linear part of the dependence of the body force $\mathbf{f}_k$ acting on phyxel k on the displacement $\mathbf{u}_l$ of phyxel l. If phyxels k and l both appear in neighborhoods of m different phyxels the submatrix $\mathbf{K}_{kl}$ is a sum of m force derivatives. For the elastic force we get

$$\begin{aligned}\frac{d}{du_l}\mathbf{f}_k = & -2v_k\frac{d}{du_l}(\mathbf{J}\sigma)\cdot\mathbf{d}_k\\ = & -2v_k\left(\frac{d}{du_l}\mathbf{J}\sigma + \mathbf{J}\frac{d}{du_l}\sigma\right)\cdot\mathbf{d}_k\\ = & -2v_k\left(\begin{bmatrix}\mathbf{d}_l^T\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}\sigma + \mathbf{J}\mathbf{C}\left(\mathbf{J}_u\mathbf{d}_l^T + \mathbf{d}_l\mathbf{J}_u^T\right)\right)\cdot\mathbf{d}_k,\end{aligned}$$

and for the volume conserving force the derivatives are

$$\begin{aligned}\frac{d}{du_l}\mathbf{f}_k = & -k_v v_k\frac{d}{du_l}\left((|\mathbf{J}|-1)\begin{bmatrix}(\mathbf{J}_v\times\mathbf{J}_w)^T\\ (\mathbf{J}_w\times\mathbf{J}_u)^T\\ (\mathbf{J}_u\times\mathbf{J}_v)^T\end{bmatrix}\right)\mathbf{d}_k\\ = & -k_v v_k\Big(\frac{d}{du_l}(|\mathbf{J}|-1)\begin{bmatrix}(\mathbf{J}_v\times\mathbf{J}_w)^T\\ (\mathbf{J}_w\times\mathbf{J}_u)^T\\ (\mathbf{J}_u\times\mathbf{J}_v)^T\end{bmatrix} + (|\mathbf{J}|-1)\frac{d}{du_l}\begin{bmatrix}(\mathbf{J}_v\times\mathbf{J}_w)^T\\ (\mathbf{J}_w\times\mathbf{J}_u)^T\\ (\mathbf{J}_u\times\mathbf{J}_v)^T\end{bmatrix}\Big)\mathbf{d}_k\\ = & -k_v v_k\Big(\begin{vmatrix}\mathbf{d}_l\\ \mathbf{J}_v^T\\ \mathbf{J}_w^T\end{vmatrix}\cdot\begin{bmatrix}(\mathbf{J}_v\times\mathbf{J}_w)^T\\ (\mathbf{J}_w\times\mathbf{J}_u)^T\\ (\mathbf{J}_u\times\mathbf{J}_v)^T\end{bmatrix} + (|\mathbf{J}|-1)\begin{bmatrix}\mathbf{0}\\ (\mathbf{J}_w\times\mathbf{d}_l)^T\\ (\mathbf{d}_l\times\mathbf{J}_v)^T\end{bmatrix}\Big)\mathbf{d}_k.\end{aligned}$$